Лабораторная работа №4. Таймер

Таймеры счетчики предназначены для:

* Подсчета временных интервалов (режим таймера);
* Подсчета числа импульсов («внешних событий») на специальном внешнем входе (режим счетчика)

Таймеры микроконтроллера функционально идентичны, за исключением их базовых периферийных адресов.

Основные характеристики таймеров

Микроконтроллер имеет в своем составе четыре 32-разрядных таймера/счетчика с 32-разрядным программируемым предделителем (рис. 6).

Предделитель

Таймер

PCLK

Прерывание

Компаратор ==

MATCH

Сброс

Рис.6. Внутренняя структура таймера в режиме сравнения

На каждый таймер имеется четыре 32-разрядных канала захвата (CAPTURE). По завершении захвата может генерироваться прерывание.

Для каждого таймера имеется четыре регистра сравнения (MATCH), которые позволяют:

* Продолжать работу при совпадении (с возможностью генерации прерывания).
* Остановить таймер при совпадении (с возможностью генерации прерывания).
* Сбросить таймер при совпадении (с возможностью генерации прерывания).

Микроконтроллер имеет четырех внешних выхода связанных с регистрами сравнения, со следующими возможностями:

* Установить низкий уровень при совпадении
* Установить высокий уровень при совпадении
* Инвертировать при совпадении

Описание регистров таймеров

Имена регистров имеют следующий вид: TnXXX, где n - номер таймера; XXX - название регистра. Например: T0IR, T1TC и т. д.

|  |  |  |
| --- | --- | --- |
| Название | Описание | Доступ |
| IR | Регистр прерываний. Если прерывание сгенерировано, то соответствующий бит в регистре IR будет установлен. Запись 1 в бит регистра IR сбросит соответствующее ему прерывание. | R/W |
| TCR | Регистр управления. Используется для управления функционированием таймера. Счетчик таймера может быть отключен или сброшен с помощью TCR. | R/W |
| TC | Счетчик таймера. 32-разрядный счетчик инкрементируется каждый PR+1 такт на линии pclk. TC управляется с помощью TCR. | R/W |
| PR | Регистр предделителя. Счетчик таймера инкрементируется каждый PR+1 такт на линии pclk. | R/W |
| PC | Счетчик предделителя. Это 32-разрядный счетчик, который инкрементируется до значения регистра PR, а затем инкрементируется TC. | R/W |
| MCR | Регистр управления сравнением. С помощью регистра MCR можно включить генерацию прерываний и сброс счетчика таймера при успешном сравнении. | R/W |
| MR0 | Регистр сравнения 0. MR0 может быть настроен с помощью регистра MCR на сброс TC, остановку TC и PC, и/или генерацию прерывания при каждом совпадении MR0 и TC. | R/W |
| MR1 | Регистр сравнения 1. Смотри описание регистра MR0. | R/W |
| MR2 | Регистр сравнения 2. Смотри описание регистра MR0. | R/W |
| MR3 | Регистр сравнения 3. Смотри описание регистра MR0. | R/W |
| CCR | Регистр управления захватом. С помощью CCR можно выбрать перепад на линии захвата, по которому происходит загрузка регистра захвата, и включить генерацию прерываний, если захват был произведен. | R/W |
| CR0 | Регистр захвата 0. CR0 загружается значением регистра TC при возникновении события на выводах CAP0.0 (CAP1.0). | RO |
| CR1 | Регистр захвата 1. Смотри описание регистра CR0. | RO |
| CR2 | Регистр захвата 2. Смотри описание регистра CR0. | RO |
| CR3 | Регистр захвата 3. Смотри описание регистра CR0. | RO |
| EMR | Регистр внешних выводов сравнения. EMR управляет внешними выводами сравнения MAT0.0-3 (MAT1.0-3). | R/W |

Доступ: R/W - Чтение/Запись; RO - Только чтение.

Регистр IR

|  |  |  |
| --- | --- | --- |
|  | Назначение | Описание |
| 0 | Прерывание MR0 | Флаг прерывания канала сравнения 0 |
| 1 | Прерывание MR1 | Флаг прерывания канала сравнения 1 |
| 2 | Прерывание MR2 | Флаг прерывания канала сравнения 2 |
| 3 | Прерывание MR3 | Флаг прерывания канала сравнения 3 |
| 4 | Прерывание CR0 | Флаг прерывания канала захвата 0 |
| 5 | Прерывание CR1 | Флаг прерывания канала захвата 1 |
| 6 | Прерывание CR2 | Флаг прерывания канала захвата 2 |
| 7 | Прерывание CR3 | Флаг прерывания канала захвата 2 |

Регистр TCR

|  |  |  |
| --- | --- | --- |
| Номер разряда | Назначение | Описание |
| 0 | Счетчик включен | 1 - счетчики таймера и предделителя включены. 0 - счетчики отключены |
| 1 | Сброс счетчика | 1 - счетчики таймера и предделителя сбрасываются по следующему положительному перепаду pclk. Счетчики находятся в состоянии сброса до тех пор, пока TCR[1] не станет нулем |

Регистр MCR

|  |  |  |
| --- | --- | --- |
| Номер разряда | Назначение | Описание |
| 0 | Прерывание по MR0 | 1 - прерывание генерируется, когда значение TC совпадает с MR0. 0 - прерывание не происходит |
| 1 | Сброс по MR0 | 1 - сброс счетчика таймера при совпадении с MR0. 0 - эта функция отключена |
| 2 | Стоп по MR0 | 1 - при совпадении TC со значением MR0, TC и PC останавливаются и TCR[0] устанавливается в ноль. 0 - эта функция отключена |
| 3 | Прерывание по MR1 | 1 - прерывание генерируется, когда значение TC совпадает с MR1. 0 - прерывание не происходит |
| 4 | Сброс по MR1 | 1 - сброс счетчика таймера при совпадении с MR1. 0 - эта функция отключена |
| 5 | Стоп по MR1 | 1 - при совпадении TC со значением MR0, TC и PC останавливаются и TCR[0] устанавливается в ноль.  0 - эта функция отключена |
| 6 | Прерывание по MR2 | 1 - прерывание генерируется, когда значение TC совпадает с MR2. 0 -прерывание не происходит |
| 7 | Сброс по MR2 | 1 - сброс счетчика таймера при совпадении с MR2. 0 - эта функция отключена |
| 8 | Стоп по MR2 | 1 - при совпадении TC со значением MR0, TC и PC останавливаются и TCR[0] устанавливается в ноль. 0 - эта функция отключена |
| 9 | Прерывание по MR3 | 1 - прерывание генерируется, когда значение TC совпадает с MR3. 0 - прерывание не происходит |
| 10 | Сброс по MR3 | 1 - сброс счетчика таймера при совпадении с MR3. 0 - эта функция отключена |
| 11 | Стоп по MR3 | 1 - при совпадении TC со значением MR0, TC и PC останавливаются и TCR[0] устанавливается в ноль. 0 - эта функция отключена |

Тактовая частота периферийных устройств PCLK устанавливается с помощью регистров PCLKSEL0 и PCLKSEL1, и для упрощения настройки микроконтроллера она может быть выбрана равной частоте генератора на кварцевом резонаторе. Для этого должны быть запрограммированы соответствующие регистры (будут использованы в дальнейшем в функции main). Через регистр SCS необходимо разрешить работу генератора на кварцевом резонаторе:

SCS |= 0x20; // Разрешить генератор с кварцевым резонатором 12МГц

while( !(SCS & 0x40) ); //Подождать стабилизации частоты генератора

При этом системная тактовая частота CCLK будет 12МГц. Затем через регистр PCLKSEL0 необходимо установить делитель частоты для таймера. Для таймера Т0 за это отвечают биты 3:2. Их значение определяется следующим образом:

|  |  |
| --- | --- |
| Биты 3:2 | Коэффициент деления |
| 00 | PCLK = CCLK/4 |
| 01 | PCLK = CCLK |
| 10 | PCLK = CCLK/2 |
| 11 | PCLK = CCLK/8 |

Следовательно, для использования частоты 12МГц без дополнительного деления необходимо выбрать комбинацию 01 и запрограммировать регистр:

PCLKSEL0 = 0x0004; // 0000 0100 - двоичный вид комбинации 01 для разрядов 3:2

В дальнейшем можно настроить таймер на генерацию прерываний с частотой 10Гц. Для этого необходимо разрешить прерывания от таймера и установить коэффициент деления 1200000 поскольку тактовая частота таймера 12000000 (12000000/1200000 = 10). Для инициализации таймера необходимо использовать соответствующую функцию:

void InitTimer0()

{

T0TCR &= ~0x01;// Запретить таймер

T0IR = 0x01; //Конфигурация регистра прерываний

T0CTCR = 0;//Конфигурация регистра управления

T0PC = 0; //Конфигурация регистра предделителя

T0MR0 = 1200000;//Конфигурация регистра сравнения

//частота PCLK = 12 МГц

//делится на T0MR0 и дает частоту прерываний 10Гц

T0MCR = (0x01 | 0x02); //Конфигурация регистра управления режима сравнения

InstallIRQ( TIMER0\_INT, (void \*)TimerHandler0, 0x03 );

}

Функция InstallIRQ() использовалась в предыдущей работе и предназначена для инициализации прерываний от различных периферийных устройств. В данном случае необходимо установить функцию прерывания от таймера TimerHandler0:

\_\_irq \_\_nested \_\_arm void TimerHandler0()

{

T0IR = 0x01;

VICADDRESS = 0; //Сброс прерывания

FIO2PIN = !FIO2PIN; //Инвертировать светодиоды

}

В данной функции при каждом вызове (возникновении прерывания 10 раз в секунду) происходит инвертирование состояния светодиодов. Таким образом, светодиоды должны мигать с частотой 5Гц.

Выполнение определенных действий непосредственно в обработчике прерываний не всегда оптимально. В таком случае можно воспользоваться дополнительной глобальной переменной – флагом события прерывания от таймера:

char T0\_event; //Глобальная переменная - флаг события прерывания от таймера

int main (void)

{

. . .//Инициализация

while (1)//Loop forever

{

if( T0\_event ) //Если прерывание случилось

{

T0\_event = 0; //Сброс флага до следующего прерывания

. . . //Делаем то, что необходимо по прерыванию от таймера

}

}

return 0;

}

\_\_irq \_\_nested \_\_arm void TimerHandler0()

{

T0IR = 0x01;

VICADDRESS = 0; //Сброс прерывания

T0\_event = 1; //Устанавливаем флаг события прерывания от таймера

}

Лабораторная работа включает в себя несколько фронтальных и индивидуальных заданий:

1. Написать программу обработки прерываний от таймера

Для этого функцию main прототипа проекта необходимо модифицировать следующим образом:

int main (void)

{

SCS |= 0x20; // Разрешить генератор с кварцевым резонатором 12МГц

while( !(SCS & 0x40) ); //Подождать стабилизации частоты генератора

PCLKSEL0 = 0x0004; // 0000 0100 - двоичный вид комбинации 01 для разрядов 3:2

// Настроить порт

SCS |= 0x01; //Разрешить быстрый ввод/вывод

FIO2DIR = 0x00FF; // Биты 0-7 порта 2 на вывод для управления светодиодами

FIO2MASK = 0; //Все разряды порта 2 работают в быстром режиме

FIO2PIN = 0x00; // Выключить светодиоды

InitVIC(); //Инициализация системы прерываний

InitTimer0(); //Инициализация таймера Т0

while (1)//Loop forever

{

//Ничего не делаем - все происходит в функции обработки прерывания

}

return 0;

}

В проект должны быть добавлены функции настройки прерываний из предыдущей работы и функция настройки таймера. В данном примере таймер настраивается на частоту 10Гц и в прерывании от таймера светодиоды меняют свое состояние. После проверки работоспособности данного примера необходимо модифицировать программу таким образом, чтобы на светодиодах был сформирован бегущий огонь с использованием прерываний от таймера.

1. Написать программу проверки состояния кнопок с подавлением дребезга контактов

Детальное описание алгоритма опроса кнопок находится в лабораторной работе номер 2. Основное отличие заключается в способе вызова функций опроса копок. Функция GetBTN1State() для кнопки 1 и аналогичные функции для других кнопок должны вызываться из обработчика прерывания таймера. При этом таймер должен быть перепрограммирован на частоту 50-100 Гц (10-20 миллисекунд) путем соответствующего изменения регистра T0MR0 в функции InitTimer0().

1. Написать программу проверки состояния энкодера

Детальное описание алгоритма проверки состояния энкодера находится в лабораторной работе номер 2. Аналогично предыдущему заданию необходимо использовать прерывание от таймера с измененным интервалом 10-20 миллисекунд

1. Написать программу регулировки яркости светодиодов с использованием широтно-импульсной модуляции

Принцип управления яркостью светодиода с помощью широтно-импульсной модуляции (ШИМ) показан на рис.7.

50%

50%

20 мс

Светодиод включен

1

2

3

Интегральный уровень яркости

Рис.7. Управления яркостью светодиода с помощью широтно-импульсной модуляции

Глаз человека перестает замечать пульсации источников излучения с частотой выше 20Гц. Если, к примеру, светодиод включен и выключен по 50% времени в течение периода (№1 на рис.7) то его яркость будет восприниматься глазом интегрально как 50% максимального уровня непрерывного свечения, а не как последовательность вспышек. Таким образом, сохраняя постоянный период, но меняя длительность включенного состояния светодиода можно плавно регулировать его яркость.

Для программной реализации ШИМ необходимо перепрограммировать таймер на интервал 1 мc. Тогда постоянный период ШИМ 20 мс будет сформирован с помощью дополнительной переменной PWM\_count – счетчика периода ШИМ, которая будет инкрементироваться в обработчике прерываний от 0 до 19. Текущее значение переменной должно проверяться и если оно меньше заданного порога (уровня яркости), то светодиод должен быть включен, а если больше, то выключен. В данном примере 4 младших светодиода включаются только при значении переменной меньше 5 (уровень яркости 5/20), а старшие 4 светодиода горят постоянно для визуального сравнения:

char T0\_event; //Глобальная переменная - флаг события прерывания от таймера

char PWM\_count;//Глобальная переменная - счетчик периода ШИМ

int main (void)

{

. . .//Инициализация

while (1)//Loop forever

{

if( T0\_event ) //Если прерывание случилось

{

T0\_event = 0; //Сброс флага до следующего прерывания

if( PWM\_count > 5 ) //Если текущее значение интервала больше порога

FIO2SET = 0xF0;//То выключаем светодиоды (0-3)

else

FIO2SET = 0xFF;//Включаем светодиоды (0-3), если еще не дошли до порога

}

}

return 0;

}

\_\_irq \_\_nested \_\_arm void TimerHandler0() //Прерывание каждую миллисекунду

{

T0IR = 0x01;

VICADDRESS = 0; //Сброс прерывания

T0\_event = 1; //Устанавливаем флаг события прерывания от таймера

PWM\_count++;//Увеличиваем счетчик периода ШИМ

if( PWM\_count > 19 )

PWM\_count = 0;//Сбрасываем счетчик периода ШИМ при достижении максимума

}

После проверки работоспособности примера необходимо модифицировать программу следующим образом:

* Яркость всех светодиодов должна непрерывно увеличиваться от 0 до максимума.
* Яркость всех светодиодов должна увеличиваться при нажатии на первую кнопку и уменьшаться при нажатии на вторую.

Далее необходимо модифицировать программу используя индивидуальное задание в соответствии с номером бригады:

1. Светодиод 0 должен увеличивать яркость, светодиод 7 уменьшать, а остальные иметь постоянный максимальный уровень
2. Светодиоды 0-3 должны увеличивать яркость, а 4-7 одновременно уменьшать
3. Светодиоды 0-3 должны увеличивать яркость до максимума и с этого момента светодиоды 4-7 должны уменьшать яркость до 0
4. Светодиод 0 должен увеличивать яркость от 0 до максимума и остаться на этом уровне. Затем тоже должен проделать светодиод 1 и т.д.
5. Светодиод 0 должен увеличивать яркость от 0 до максимума, после чего уменьшить до 0 и остаться на этом уровне. Затем тоже должен проделать светодиод 1 и т.д.
6. Светодиод 0 должен увеличивать яркость от 0 до 50% и в этот момент светодиод 1 должен начать увеличивать яркость от 0 до максимума.
7. Светодиод 0 должен увеличивать яркость, а 1 одновременно уменьшать. По достижении предела поменять направление изменения яркости.
8. Светодиод 0 должен увеличивать яркость от 0 до 25%, а светодиод 1 до 50%
9. Светодиод 0 должен увеличивать яркость от 25 до 50%, а светодиод 1 от 50% до 75%
10. Светодиод 0 должен увеличивать яркость от 0 до 50%, а светодиод 1 уменьшать от максимума до 50%
11. Светодиод 0 должен менять яркость от 25% до 75%, а светодиод 1 от 75% до 25%
12. Светодиоды 0-1 должны менять яркость от 0 до 50%, а затем светодиод 1 до максимума